2 1 Best for The case

First Hit Fwd Refs

Generate Collection Print

L22: Entry 3 of 17 File: USPT Dec 12, 2000

DOCUMENT-IDENTIFIER: US 6161160 A
TITLE: Network interface device architecture for storing transmit and receive data in a random access buffer memory across independent clock domains

Abstract Text (1):
A network interface device includes a random access transmit buffer and a random access receive buffer for transmission and reception of transmission and receive data frames between a host computer bus and a packet switched network. The network interface device includes a memory management unit having read and write controllers for each of the transmit and receive buffers, where each write controller operates in a clock domain separate from the corresponding read controller. The memory management unit also includes a synchronization circuit that controls arbitration for accessing the random access memories between the read and write controllers. The synchronization circuit asynchronously monitors the amount of data stored in the random access transmit and receive buffer by asynchronously comparing write pointer and read pointer values stored in gray code counters, where each counter is configured for changing a single bit of a counter value in response to an increment signal. A descriptor management unit is used to control DMA reading and writing of transmit data and receive data from and to system memory, respectively, based on descriptor lists, respectively. A pipelining architecture also optimizes transfer of data between the buffers, the PCI bus, and the media access controller.

Brief Summary Text (5):
Network interface devices handle packets of data for transmission between a host computer and a network communications system, such as a local area network. The host computer may be implemented as a client station, a server, or a switched hub. One primary function of the network interface device is to buffer data to compensate for timing discrepancies between the clock domain of the host computer and the clock domain of the network.

Brief Summary Text (6):
Network interface devices typically include a first in, first out (FIFO) buffer memory for storing transmit and receive data, where the transmit data is stored in a transmit FIFO prior to transmission on the network media by the MAC, and receive data is stored in a receive FIFO by the MAC prior to transfer to the host computer via the host computer bus interface.

Brief Summary Text (7):
One disadvantage with the use of a FIFO for a transmit buffer or a receive buffer is the increased latency encountered during the buffering process. The latency of the network interface device is the time delay between the time that a data frame is supplied to the network interface device and the time the data is transmitted on the network media, or vice versa.

Brief Summary Text (8):
An additional disadvantage with the use of a FIFO for transmit buffer or receive buffer is the increasing complexity associated with maintaining status information for each data frame stored in the FIFO buffer. If a stored data frame is to have corresponding status information, then an additional FIFO buffer would be required

for storing the status information for each stored data frame. Hence, a transmit buffer may require a data frame FIFO for the actual frame data, and a status FIFO for storing the corresponding status information for each data frame. Such an arrangement would result in a substantial increase in the amount of area required on a chip for the status FIFO. In addition, additional synchronization logic would be required to maintain correspondence between the stored frame data and the corresponding status data, increasing the cost and complexity of the network interface device.

Brief Summary Text (9):
An additional problem caused by the buffering of data between the clock domain of the host computer and the clock domain of the network interface device is buffer overflow or underflow. For example, buffer overflow can occur when the time domains between the host bus and the network media are uncontrollable to the extent that data is stored in the buffer at a rate faster than the data can be removed, resulting in an overflow situation. Conversely, underflow occurs if data is removed from the FIFO buffer faster than the data can be supplied.

Brief Summary Text (10):
Hence, the non-synchronous relationship between the host bus clock domain and the network clock domain have required the necessity of FIFO buffers to compensate for timing discrepancies between the host computer and the network.

Brief Summary Text (11):
Another fundamental problem with use of a FIFO as a transmit buffer or receive buffer is that there is no convenient way for the network interface device to bypass, or "flush," invalid data. For example, if the media access controller receives a runt packet from the network (i.e., an invalid packet less than the minimum required frame size of 64 bytes), the MAC cannot cause the invalid data stored in the FIFO to be flushed, without eliminating the entire contents of the receive FIFO. Hence, the invalid data is transferred via the host computer bus and stored in host computer memory, before the host computer can determine that the transferred data is invalid. The reduction in throughput may have substantial effects, especially in full-duplex networks, where the host computer bus is heavily utilized by the network interface device for simultaneous transmission and reception of data frames on the network medium.

Brief Summary Text (14):
An additional problem with conventional network interface devices is the occurrence of wait states encountered during a complex bus termination condition, where certain events on the PCI bus forcibly halt a PCI bus data transfer. Two examples of complex conditions include when a host memory is not ready to receive a data transfer after the bus has been secured, or when the host memory becomes unable to continue receiving data following initiation of the data transfer. In either case, the target asserts a STOP# signal on the PCI bus to terminate the data transfer. Prior art systems frequently lose data from the FIFO buffer memory in response to encountering such complex conditions, for example retry or disconnect states. Hence, complicated recovery arrangements are conventionally required in prior art systems to mitigate the loss of data. For example, higher network protocol layers may need to send a message across the network, requesting the transmitting station to resend a data packet.

Brief Summary Text (16):
There is a need for an arrangement that enables use of a random access memory in a network controller, as opposed to a FIFO buffer, to compensate for timing discrepancies between the host computer and the network.

Brief Summary Text (17):
There is also a need for an arrangement enabling the use of a random access memory as a buffer in a network interface device, where potential synchronization problems

between the clock domain of the host computer and the clock domain of the network are resolved to enable efficient control of the random access memory during the writing and reading of transmit or receive data.

Brief Summary Text (19):
There is also a need for an arrangement in a network interface device, where a network interface architecture asynchronously monitors the status of data stored in a random access transmit buffer and a random access receive buffer to enable multiple memory controllers to store and read data into the random access memories using multiple clock domains.

Brief Summary Text (20):
There is also a need for an arrangement in a network interface device having a memory controller that enables access to a random access transmit buffer or a random access receive buffer according to either a direct memory access (DMA) or slave access.

Brief Summary Text (21):
There is also a need for an arrangement in a network interface device having a circuit that asynchronously monitors the status of a random access transmit buffer and a random access receive buffer, enabling memory controllers to operate in response to prescribed conditions detected by the synchronization circuit.

Brief Summary Text (22):
These and other needs are attained by the present invention, where a memory management unit, configured for controlling transfer of transmit data and receive data into respective random access transmit and receive buffers, includes a synchronization circuit for asynchronously monitoring the amount of data stored in the random access transmit buffer and the random access receive buffer. The asynchronous monitoring by the synchronization circuit enables memory management unit operations to be performed in respective independent clock domains.

Brief Summary Text (23):
According to one aspect of the present invention, a method in a network interface device for sending data frames from a host computer to a network medium comprises storing transmit data received from a host bus into a random access transmit buffer according to a host bus clock, asynchronously monitoring the amount of data stored in the random access transmit buffer, and outputting the stored transmit data from the random access transmit buffer to a media access controller according to a transmit clock independent from the host bus clock and based on the asynchronously monitoring step, for transmission on the network medium. The asynchronous monitoring of the amount of data stored in the random access transmit buffer enables the transmit data to be stored in the random access transmit buffer according to the host bus clock domain, and also enables the transmit data to be output from the random access transmit buffer according to a network transmit clock, independent from the host bus clock, with minimal latency in the network interface device. Since the amount of data stored in the random access transmit buffer is monitored asynchronously, operations in the network interface device may be performed on an event-driven basis, where the asynchronous detection of events within the random access transmit buffer enables optimized performance in the appropriate clock domain.

Brief Summary Text (24):
Another aspect of the invention provides a method in a network interface device for receiving data frames from a network medium to a host computer, comprising storing receive data received from a media access controller into a random access receive buffer according to a network receive clock, asynchronously monitoring the amount of data stored in the random access receive buffer, and outputting the stored receive data from the random access receive buffer to a host bus interface according to a host bus clock independent from the network receive clock and based

on the asynchronously monitoring step, for transmission on a host bus.

Brief Summary Text (25):
Still another aspect of the present invention provides a network interface device comprising a media access controller configured for simultaneously outputting transmit data according to a network transmit clock, and receiving receive data from a network medium according to a network receive clock, a bus interface unit configured for transferring via a host bus the receive data and the transmit data to and from a host computer memory according to a host bus clock, a random access receive buffer configured for storing the receive data received by the media access controller according to the network receive clock, and outputting the stored receive data to the bus interface unit according to the host bus clock, a random access transmit buffer configured for storing the transmit data supplied by the bus interface unit according to the host bus clock, and outputting the stored transmit data to the media access controller according to the network transmit clock, and a memory management unit. The memory management unit is configured for controlling the transfer of the transmit data and receive data in the random access transmit buffer and the random access receive buffer, and includes a synchronization circuit for asynchronously monitoring the amount of data stored in the random access transmit buffer and the random access receive buffer. The asynchronous monitoring by the synchronization circuit enables the memory management unit to interact with the media access controller, the bus interface unit, and the random access transmit and receive buffers on an event-driven basis, where operations can be performed in the appropriate clock domain based on the asynchronous monitoring of events in the synchronization circuit.

Brief Summary Text (26):
This aspect of the network interface device is particularly advantageous in the case where the memory management unit includes first, second, third, and fourth management blocks configured for controlling the transfer of transmit data to and from the random access transmit buffer, and receive data to and from the random access receive buffer, respectively. Hence, each of the management blocks can be optimized for performing its corresponding memory functions in its corresponding clock domain by obtaining any relevant status information from the synchronization circuit. Hence, the network interface device can be configured for efficient transfer of transmit and receive data, in a manner that eliminates any contention issues between the host bus clock, the network transmit clock or the network receive clock.

Drawing Description Text (4):
FIGS. 1A and 1B illustrate an exemplary network interface device including a synchronization circuit for controlling buffer memory controllers according to an embodiment of the present invention;

Drawing Description Text (5):
FIG. 2 is a block diagram illustrating the buffer architecture of the network interface device of FIG. 1 according to an embodiment of the present invention.

Drawing Description Text (10):
FIG. 7 is a diagram illustrating a method of transferring frame data by the descriptor management 18 between system memory and the random access buffer memories.

Drawing Description Text (11):
FIGS. 8A, 8B, 8C, and 8D are block diagrams illustrating holding registers used to buffer data input to the transmit random access memory, output from the transmit random access memory, input to the receive random access buffer memory, and output from the receive random access buffer memory, respectively.

Detailed Description Text (2):

The present invention will be described with the example of a network interface device in a packet switched network, such as an Ethernet (IEEE 802.3) network. A description will first be given of a network interface architecture, followed by the arrangement for monitoring the storage of a data frame in a buffer memory, independent of host computer clock and network data clock domains. It will become apparent, however, that the present invention is also applicable to other network interface device systems.

Detailed Description Text (6):
The interface 10 includes a PCI bus interface unit 16, a buffer memory portion 18, and a media access controller interface device (MAC) 20. The PCI bus interface unit 16 includes a PCI slave interface 16a and a DMA interface 16b. The slave interface 16a manages PCI control and status information including reading and programming of the PCI status registers, but may also be configured for managing slave transfers via the PCI bus with a host CPU. The DMA interface 16b manages DMA transfers by the network interface device 10 to and from system memory. Hence, the PCI bus interface unit 16 can be selectively configured for PCI transfers in slave and/or master (e.g., DMA) mode.

Detailed Description Text (8):
The network interface device 10 also includes a buffer management unit 24 configured for managing DMA transfers via the DMA interface 16b. The buffer management unit 24 manages DMA transfers based on DMA descriptors in host memory that specify start address, length, etc. The buffer management unit 24 initiates a DMA read from system memory into the transmit buffer 18b by issuing an instruction to the DMA interface 16b, which translates the instructions into PCI bus cycles. Hence, the buffer management unit 24 contains descriptor management for DMA transfers, as well as pointers associated with storing and reading data from the memory portion 18. Although the buffer management unit 24 and the memory management unit 22 are shown as discrete components, the two units may be integrated to form a memory management unit controlling all transfers of data to and from the memory unit 18.

Detailed Description Text (15):
FIG. 2 is a block diagram illustrating the buffer architecture of the network interface device 10 according to an embodiment of the present invention. As shown in FIG. 2, transfer of data frames between the PCI bus interface unit 16, also referred to as the bus interface unit (BIU), and the MAC 20 is controlled by a memory management unit (MMU) 52 including the buffer management unit 24 and the SRAM MMU 22 of FIG. 1. The MMU 52 controls the reading and writing of data to the SRAM 18, illustrated in FIG. 2 as a dual-port receive SRAM portion 18a and a dual-port transmit SRAM portion 18b for convenience. It will be recognized in the art that the receive SRAM (RX.sub.-- SRAM) 18a and the transmit SRAM (TX.sub.-- SRAM) 18b may be implemented as a single memory device, or alternatively as two separate SRAM devices.

Detailed Description Text (16):
As shown in FIG. 2, the memory management unit includes the buffer management unit 24, also referred to as the descriptor management unit, the SRAM MMU 22, and an arbitration unit 54. The descriptor management unit 24 is configured for fetching, from host computer memory, descriptor information (i.e., transfer information) specifying host computer memory locations for retrieving and storing transmit data and receive data, respectively. The arbitration unit 54 arbitrates DMA requests for data transmission, data reception, descriptor lists from the descriptor management block 24, and status.

Detailed Description Text (22):
FIG. 5A is a diagram illustrating multiple data frames (F1, F2, etc.) stored in the RX.sub.-- SRAM 18a. Assume that the RM.sub.-- MMU 22d is writing a sequence of data frames 64 (frame 1, frame 2, etc.) into RX.sub.-- SRAM 18a using a write pointer

(WP), while the read controller 22c is reading out the data frames from the RX.sub.-- SRAM 18a to the BIU 16 using a read pointer (RP). If the read controller discards (e.g., flushes) a transmit data frame and desires to jump to the beginning of the next data frame, the synchronization circuit 60 must be able to track the start and beginning of each data frame to ensure that the read controller 22c properly locates the beginning of the next data frame.

Detailed Description Text (23):
FIG. 4 is a block diagram illustrating in detail the MMU 22. The synchronization circuit 60 includes asynchronous monitors 82a and 82b for asynchronously monitoring the amount of stored receive data and transmit data in the SRAMs 18a and 18b respectively, enabling the memory management units 22a, 22b, 22c, and 22d to track the number of stored data frames in their respective clock domains. The memory management unit tracks the number of stored data frames based on the difference between the write frame counter value and the read frame counter values. For example, the management block 22a includes a write counter 84a for incrementing a write pointer value (Tx Write Ptr.) in response writing the transmit data to the random access transmit buffer 18b. The management block 22b includes a read counter 84b for incrementing a read pointer value (Tx Read Ptr.) in response to reading the transmit data from the random access transmit buffer 18b. Similarly, the management block 22d includes a write counter 84d for incrementing a write pointer value (Rx Write Ptr.) in response to writing the receive data to the random access receive buffer 18a, and the management block 22c includes a read counter 84c for incrementing a read pointer value (Rx Read Ptr.) in response to reading the receive data from the random access receive buffer 18a. As described above, the XB.sub.-- MMU 22a and the RB.sub.-- MMU 22c operate in the PCI bus clock domain (BCLK) 56a, whereas the XM.sub.-- MMU 22b and the RM.sub.-- MMU 22d operate in the network clock domain 56b. More specifically, the XM.sub.-- MMU 22b operates in the network transmit clock domain (XMCLK), and the RM.sub.-- MMU operates in the network receive clock domain (RMCLK).

Detailed Description Text (25):
For example, assume a read frame counter pointer value (Rx Read Ptr.) and a write counter pointer value (Rx Write Ptr.) are stored in binary counters, where a write counter has a value (WR=100) and a read counter in the second independent clock domain transitions from (RD=011) to (RD=100). Since the clock domains 56a and 56b operate independently of each other, a logic comparator performing a comparison between the write counter and read counter may erroneously conclude that the read and write counters have different values at a point in time where the read counter has a transitional value (e.g., 101, 111, or 000) as the read counter is being updated. Hence, the attempt to perform an asynchronous comparison between the binary read and write counters may cause an erroneous conclusion that the read and write pointers are not equal.

Detailed Description Text (26):
One possible solution for preventing asynchronous comparisons during counter transitions is to provide latched outputs for the counter values. However, such an arrangement would severely degrade the timing performance of the random access memory as a buffer device.

Detailed Description Text (34):
Hence, the synchronization circuit can asynchronously determine the presence of at least one stored data frame in either the RX SRAM 18a or the TX SRAM 18b, independent of the clock domains used to write and read into the respective transmit or receive buffers. As described below, the use of gray code counters may also be implemented in each of the management blocks 22a, 22b, 22c, and 22d to enable the synchronization circuit 60 to asynchronously determine the amount of transmit data and receive data in the transmit SRAM 18b and receive SRAM 18a, respectively.

Detailed Description Text (38):
According to the disclosed embodiment, the synchronization circuit 60 selectively stores frame track information based on the asynchronously determined presence of at least one data frame in the corresponding transmit or receive buffer. For example, the synchronization circuit 60 selectively stores either the first tracking information (RM.sub.-- FRM.sub.-- TRK) or the second tracking information (RB.sub.-- FRM.sub.-- TRK) to the holding register 86a based on an asynchronous detection of at least one stored data frame in the receive memory 18a. The synchronization circuit 60 outputs a one-or-more received frame signal (RX.sub.-- FRM.sub.-- ONEM) equal to a value of 1 if the RX.sub.-- SRAM 18a stores at least one data frame as shown in FIG. 5A. If the memory 18a stores less than a complete frame of data, as shown in FIG. 5B, then the synchronization circuit 60 outputs the one-or-more signal as having a value of RX.sub.-- FRM.sub.-- ONEM=0. Hence, priority is given to the read controller 22c to write the corresponding frame track information into the holding register 86a if one or more frames are stored in memory 18a as shown in FIG. 5A, and grants priority to the write controller 22d if less than one full frame (i.e., a complete frame) is stored in memory 18a, as shown in FIG. 5B.

Detailed Description Text (40):
As described above, the synchronization circuit 60 asynchronously monitors the amount of data stored in the random access transmit buffer 18b and the random access receive buffer 18a, enabling each of the management blocks 22a, 22b, 22c, and 22d to operate as event-driven controllers that transfer data in response to detected conditions by the synchronization circuit 60. Hence, each of the management blocks 22a, 22b, 22c, and 22d can operate in its corresponding clock domain, without concern of operations in other management blocks operating in other clock domains, by obtaining relevant status conditions from the synchronization circuit 60.

Detailed Description Text (41):
As shown in FIG. 4, each management block 22.sub.i includes a memory counter for incrementing a memory pointer value in response to reading or writing the corresponding data from the corresponding SRAM. For example, the XB.sub.-- MMU 22a includes a transmit write counter 84a for incrementing a write pointer value in response to writing a double word (DWORD) into a memory location of the 32-bit SRAM 18b. According to the disclosed embodiment, each of the counters 84 are implemented as gray code counters, where the counter changes a single bit of the corresponding pointer value in response to the writing or reading of the corresponding data. Hence, the XB.sub.-- MMU 22a increments the write counter 84a by changing a single bit of the TX Write Ptr. value in response to writing a DWORD of transmit data into the TX.sub.-- SRAM 18b. Similarly, the XM.sub.-- MMU 22b includes a read counter 84b that increments the read pointer value (TX Read Ptr.) by changing a single bit of the corresponding read point value in response to reading a DWORD from the random access transmit buffer 18b. As shown in FIG. 4, the RX write counter 84d and the RX read counter 84c operate in the same manner, i.e., by changing a single bit of the corresponding pointer value in response to writing or reading a DWORD from the RX.sub.-- SRAM 18a.

Detailed Description Text (42):
As shown in FIG. 4, each of the management blocks 22a, 22b, 22c, and 22d output the corresponding pointer value to the synchronization circuit 60. Specifically, the RX Write Ptr. value and the RX Read Ptr. value are supplied to an asynchronous monitor 82a that asynchronously determines the amount of stored receive data (e.g., on a double word basis) in the RX.sub.-- SRAM 18a. For example, the monitor 82a may include a comparator for performing a gray code comparison of the read and write pointer values. Hence, the asynchronous comparator 82a asynchronously monitors the amount of data stored in the random access receive buffer 18a.

Detailed Description Text (43):

Similarly, the asynchronous monitor 82b asynchronously monitors the amount of data stored in the random access transmit buffer 18b by performing a gray code comparison between the TX Write Ptr. value and the TX Read Ptr. value from the counters 84a and 84b, respectively.

Detailed Description Text (48):
FIG. 6 is a diagram illustrating descriptor lists in system memory, used by the descriptor management block 24 to obtain information related to retrieval and storage of transmit data and receive data into system memory, respectively. The descriptors 100 identify locations 102 in system memory 104 that store at least portions 106 of a data frame, respectively. In particular, each descriptor 100 specifies the starting address (e.g., TX.sub.-- BUF.sub.-- ADDR) of where the descriptor management 24 may find the transmit data portion 106 in system memory. The descriptor 100 also includes control information, the buffer length, etc., enabling the network interface device 10 to perform a DMA transfer by the PCI bus. After the descriptor management 24 fetches the descriptor information 100, the descriptor management 24 will write status information back to system memory upon completion of the data transfer (e.g., reading transmit data from system memory, or writing receive data to system memory).

Detailed Description Text (49):
FIG. 7 is a flow diagram illustrating operations by the descriptor management 24 in controlling the transfer of frame data between the system memory 104 and the buffer memory 18. As shown in FIG. 7, the descriptor management receives from the host computer descriptor index information that specifies the location in system memory of the descriptors 100. For example, the host CPU may first write the frame data 106 into the system memory 104, then write the corresponding descriptors 100 in a second location in system memory. The host computer will then write to a control register in the descriptor management 24 addressable by the host CPU on the PCI bus, that identifies the location of the descriptors 100 and releases control of the descriptors from the host CPU to the network interface device 10.

Detailed Description Text (51):
The arbiter 54 returns a grant to the descriptor management 24 and in response generates a PCI bus request signal (REQ#) on the PCI bus. Once the PCI bus arbiter (e.g., the host CPU) generates a grant (GNT#), the descriptor management 24 in response fetches the descriptor information 100 in step 202.

Detailed Description Text (52):
As shown in FIG. 6, each descriptor 100 includes a 64-bit system address (e.g., TX.sub.-- BUF.sub.-- ADDR) that points to a corresponding system memory location 102.sub.i. The descriptor management also determines the length of the corresponding frame data fragment 106 by reading the corresponding length from the buffer length field (BUF LENGTH). After decoding the descriptor information in step 204, the descriptor management 24 initiates a second PCI bus transfer in step 206 to transfer the frame data 106.sub.i between the system memory 104 and the appropriate SRAM buffer 18.

Detailed Description Text (53):
Following transfer of the frame data 106 in step 206 (e.g., burst transferring the transmit data 106 from the host computer memory 106 to the random access transmit buffer 18b), the descriptor management 24 issues a third PCI bus transfer in step 208 to write status information to the host computer memory based on the successful burst transfer in step 206, releasing control of the descriptors 100 back to the driver software executed by the host CPU.

Detailed Description Text (54):
Although FIG. 6 is disclosed with respect to the transmit buffer in system memory, the above-described procedures in FIGS. 6 and 7 is identical for receive data, where the descriptor management 24 fetches descriptor information from system

memory that specifies the locations in system memory where the network interface device 10 is to transfer the receive data from the receive SRAM 18a to the system memory 104. Hence, the descriptor management follows the same sequence of fetching descriptor information, burst transferring the stored receive data to the destination via the host bus based on the amount of stored receive data in the RX.sub.-- SRAM and the corresponding descriptors, and writing status information to the host computer memory based on the burst transfer of the stored receive data to the destination. According to the disclosed embodiment, the descriptor management 24 writes a single double word (DWORD) of status information for transmit data, and two DWORDs of status information for receive data.

Detailed Description Text (55):
Hence, the descriptor management 24 maintains a certain number of descriptors in the network interface device 10 as needed to transfer transmit data and receive main data between the system memory and the random access buffer memories 18. Hence, the amount of memory space on the network interface device needed to control data transfers is minimized. Further, the disclosed arrangement provides an efficient protocol with the driver software executed by the host CPU, minimizing contention conditions for determining whether certain descriptors 100 are to be accessed by the network interface device 10 or the driver software.

Detailed Description Text (57):
As shown in FIG. 8A, the bus interface unit 16 includes a D flip flop 210a that latches the data (XB.sub.-- DATA) from the AD signal path of the PCI bus 12 in response to the host bus clock (BCLK). The D flip flop 210a outputs the 32-bit XB.sub.-- DATA signal to the holding register 212b, which selectively latches the XB.sub.-- DATA in response to an XB.sub.-- ADV signal received from the BIU 16. According to the disclosed embodiment, the multiplexer of holding register 212b is a multiple-selection input multiplexer, such that byte packing may be performed on the received XB.sub.-- DATA. Hence, the data output by the holding register 212b (XB.sub.-- SRAM.sub.-- DATA) is aligned in the TX.sub.-- SRAM to avoid the presence of invalid data within the TX.sub.-- SRAM due to invalid byte lanes (i.e., "holes"). The holding register 212a outputs the write address pointer (XB.sub.-- ADDR), as controlled by the incrementer 214a, controlled by the XB.sub.-- MMU 22a. As shown in FIG. 8A, the supply of data from the PCI bus 12 is performed in the host bus (BCLK) domain. Hence, the XB.sub.-- ADV signal from the BIU 16 enables the XB.sub.-- MMU to increment the write pointer in holding register 212a, and to supply data to the dual-port TX.sub.-- SRAM 18b.

Detailed Description Text (59):
As shown in FIG. 8B, the incrementer 214b, under the control of the XM.sub.-- MMU 22b, controls the read address pointer stored in holding register 212d. The MAC 20 outputs an advance signal (XM.sub.-- ADV), used to increment the read pointer in holding register 212d and the read signal in holding register 212e. According to the disclosed embodiment, the MAC 20 is able to assert the XM.sub.-- ADV advance signal until the TX.sub.-- SRAM is empty, based on an empty flag TX.sub.-- SRAM.sub.-- registers 212f and 212g are part of the MAC 20.

Detailed Description Text (61):
FIG. 8D is a diagram illustrating in detail the data path for the RB.sub.-- MMU 22c. According to the disclosed embodiment, the RB.sub.-- MMU 22c outputs the data onto the AD signal path of the PCI bus 12 with zero wait states in a DMA transfer, based on the holding registers 212l, 212m, 212n, and the multiplexer 216. As shown in FIG. 8D, the advance signal RB.sub.-- ADV output by the BIU 16 is used to drive the read pointer stored in holding register 212j, and the read pointer stored in holding register 212k. As described above, the read pointer value stored in holding register 212j is controlled by the incrementer 214c. Hence, the RX.sub.-- SRAM 18a outputs a double word of data from a prescribed location pointed to by the read pointer value (RB.sub.-- ADDR) as a 32-bit signal RB.sub.-- SRAM.sub.-- DATA. The holding register 212l outputs the signal RB.sub.-- DATA based on the RB.sub.-- ADV

signal and the bus clock (BCLK) signal. Although not shown, the holding register 2121 also performs byte alignment relative to the PCI bus 12 based on byte alignment (BE.sub.-- L) signals received by the multiplexer 220 from the descriptor management 24. The holding register 2121 outputs the RB.sub.-- DATA signals to the holding register 212m and the multiplexer 216. The holding register 212m is configured to resolve any wait states or bus termination conditions (e.g., disconnect, retry) that may occur on the PCI bus 12. The multiplexer 216 and the multiplexer of the holding register 212n are controlled by a state machine (not shown) in the bus interface unit.

Detailed Description Text (63):
According to the disclosed embodiment, the memory management unit provides an efficient manner for reading and writing frame data from a transmit random access buffer memory and a receive random access buffer memory, despite the presence of multiple clock domains. The use of holding registers also enables the efficient pipelining of data, such that zero wait states are encountered during a DMA burst write, and minimal delay is encountered during retransmission in response to a detected disconnect or retry condition on the PCI bus.

CLAIMS:

1. A method in a network interface device for sending data frames from a host computer to a network medium, comprising:

storing transmit data received from a host bus into a random access transmit buffer according to a host bus clock;

asynchronously monitoring the amount of data stored in the random access transmit buffer; and

outputting the stored transmit data from the random access transmit buffer to a media access controller according to a transmit clock independent from the host bus clock and based on the asynchronously monitoring step, for transmission on the network medium.

2. The method of claim 1, wherein the random access transmit buffer includes an input port and an output port, the storing step comprising supplying the transmit data to the random access transmit buffer via the input port.

4. The method of claim 1, wherein the asynchronously monitoring step comprises:

incrementing a write pointer value in response to the storing step;

incrementing a read pointer value in response to the outputting step; and

asynchronously comparing the write pointer value and the read pointer value to determine an amount of stored data in the random access transmit buffer.

6. The method of claim 5, wherein the outputting step comprises:

outputting a second signal from the media access controller in response to the first signal and in response to a detected idle condition on the network medium; and

supplying the stored transmit data from the random access transmit buffer to the media access controller in response to the second signal.

8. The method of claim 7, wherein:

the asynchronously monitoring step further comprises selectively storing in a

holding register information identifying the amount of transmit data corresponding to said data frame in response to said storing step and the host bus clock, and based on a determined presence of at least one data frame in the random access transmit buffer;

the asynchronously determining step comprises reading the information identifying the amount of transmit data corresponding to said data frame from the holding register based on the transmit clock.

9. The method of claim 1, further comprising:

storing receive data received from the media access controller into a random access receive buffer according to a receive clock independent from the host bus clock and the transmit clock;

asynchronously monitoring the amount of receive data stored in the random access receive buffer; and

outputting the stored receive data from the random access receive buffer onto the host bus according to a the host bus clock and based on the step of asynchronously monitoring the amount of stored receive data.

10. The method of claim 9, wherein:

the random access transmit and the random access receive buffers each comprise an input port and an output port;

the transmit data storing step comprises supplying the transmit data to the input port of the random access transmit buffer according to the host bus clock; and

the stored transmit data outputting step comprises supplying the transmit data to the media access controller via the output port of the random access transmit buffer according to the transmit clock.

11. The method of claim 10, wherein:

the receive data storing step further comprises supplying the receive data to the input port of the random access receive buffer according to the receive clock; and

the stored receive data outputting step further comprises supplying the receive data to the host bus via the output port of the random access receive buffer according to the host bus clock.

13. The method of claim 9, wherein the media access controller has transmit and receive ports, the receive data storing step comprising supplying the receive data to the random access receive buffer via the receive port independent of the stored transmit data outputting step via the transmit port.

14. The method of claim 9, further comprising:

retrieving first descriptor information from the host computer memory specifying a destination for the stored receive data;

burst transferring the stored receive data to the destination via the host bus based on the amount of stored receive data in the random access receive buffer; and

writing status information to the host computer memory based on the burst transfer of the stored receive data to the destination.

15. The method of claim 14, further comprising:

retrieving second descriptor information from the host computer memory specifying a source for the transmit data in the host computer memory;

burst transferring the transmit data from the source to the random access transmit buffer based on the descriptor information and the amount of data stored in the random access transmit buffer; and

writing status information to the host computer memory based on the burst transfer of the transmit data to the host computer memory.

16. The method of claim 15, further comprising prioritizing between the burst of the receive data and the burst transferring of the receive data based on the amounts of data stored in the random access receive buffer and the random access transmit buffer, respectively.

17. A method in a network interface device for receiving data frames from a network medium, comprising:

storing receive data received from a media access controller into a random access receive buffer according to a network receive clock;

asynchronously monitoring the amount of data stored in the random access receive buffer; and

outputting the stored receive data from the random access receive buffer to a host bus interface according to a host bus clock independent from the network receive clock and based on the asynchronously monitoring step, for transmission on a host bus.

18. The method of claim 17, wherein the random access receive buffer includes an input port and an output port, the storing step comprising supplying the receive data to the random access receive buffer via the input port based on the monitoring step.

20. The method of claim 17, wherein the asynchronously monitoring step comprises:

incrementing a write pointer value in response to the storing step;

incrementing a read pointer value in response to the outputting step; and

asynchronously comparing the write pointer value and the read pointer value to determine the amount of data stored in the random access receive buffer.

22. A network interface device, comprising:

a media access controller configured for simultaneously outputting transmit data according to a network transmit clock, and receiving receive data from a network medium according to a network receive clock;

a bus interface unit configured for transferring via a host bus the receive data and the transmit data to and from a host computer memory according to a host bus clock;

a random access receive buffer configured for storing the receive data received by the media access controller according to the network receive clock, and outputting the stored receive data to the bus interface unit according to the host bus clock;

a random access transmit buffer configured for storing the transmit data supplied

by the bus interface unit according to the host bus clock, and outputting the stored transmit data to the media access controller according to the network transmit clock; and

a memory management unit configured for controlling the transfer of the transmit data and receive data in the random access transmit buffer and the random access receive buffer, the memory management unit comprising a synchronization circuit for asynchronously monitoring the amount of data stored in the random access transmit buffer and the random access receive buffer.

23. The network interface device of claim 22, wherein the memory management unit further comprises:

a first management block for controlling the transfer of the transmit data from the host computer memory to the random access transmit buffer according to the host bus clock and based on the amount of data specified by the synchronization circuit; and

a second management block for controlling the transfer of the stored transmit data from the random access transmit buffer to the media access controller according to the network transmit clock and based on the amount of data specified by the synchronization circuit.

24. The network interface device of claim 23, wherein:

the first management block includes a write counter for incrementing a write pointer value in response to writing the transmit data to the random access transmit buffer;

the second management block includes a read counter for incrementing a read pointer value in response to reading the transmit data from the random access transmit buffer; and

the synchronization circuit asynchronously determines the amount of data stored in the random access transmit buffer based on the write pointer value and the read pointer value.

25. The network interface device of claim 24, wherein the write counter and the read counter each increment the corresponding pointer value by changing a single bit of the corresponding pointer value.

26. The network interface device of claim 23, wherein the memory management unit further comprises:

a descriptor management unit configured for fetching, from the host computer memory, transfer information specifying locations in the host computer memory for at least one of the receive data and the transmit data; and

an arbitration unit for selectively causing the transfer of one of the transmit data and the receive data on the host bus based on the amount of data stored in the random access transmit buffer and the random access receive buffer.

28. The network interface device of claim 23, wherein the memory management unit further comprises:

a third management block for controlling the transfer of the receive data from the media access controller to the random access receive buffer according to the network receive clock and based on the amount of data stored in the random access receive buffer specified by the synchronization circuit; and

a fourth management block for controlling the transfer of the stored receive data from the random access receive buffer to the host computer memory according to the host bus clock and based on the amount of data stored in the random access receive buffer specified by the synchronization circuit.

30. The network interface device of claim 29, wherein the memory management unit further comprises:

a descriptor management unit configured for fetching, from the host computer memory, transfer information specifying locations in the host computer memory for at least one of the receive data and the transmit data; and

an arbitration unit for selectively causing the transfer of one of the transmit data and the receive data on the host bus based on the amount of data stored in the random access transmit buffer and the random access receive buffer.

33. The network interface device of claim 29, wherein the third management block includes at least one register configured for storing a prescribed amount of the receive data for a prescribed successive number of network receive clock cycles, and outputting the prescribed amount of data to the random access receive buffer in a single network receive clock cycle.

35. The network interface device of claim 23, wherein the first management block includes at least one holding register configured for selectively supplying bytes of the transmit data from the host bus to the random access transmit buffer.

36. The network interface device of claim 23, wherein the second management block includes at least one holding register configured for selectively supplying bytes of the stored transmit data from the random access transmit buffer to the media access controller based on an advance signal from the media access controller.

this is good to

2

L47: Entry 5 of 20 File: USPT Dec 12, 2000

DOCUMENT-IDENTIFIER: US 6161160 A
TITLE: Network interface device architecture for storing transmit and receive data in a random access buffer memory across independent clock domains

Abstract Text (1):
A network interface device includes a random access transmit buffer and a random access receive buffer for transmission and reception of transmission and receive data frames between a host computer bus and a packet switched network. The network interface device includes a memory management unit having read and write controllers for each of the transmit and receive buffers, where each write controller operates in a clock domain separate from the corresponding read controller. The memory management unit also includes a synchronization circuit that controls arbitration for accessing the random access memories between the read and write controllers. The synchronization circuit asynchronously monitors the amount of data stored in the random access transmit and receive buffer by asynchronously comparing write pointer and read pointer values stored in gray code counters, where each counter is configured for changing a single bit of a counter value in response to an increment signal. A descriptor management unit is used to control DMA reading and writing of transmit data and receive data from and to system memory, respectively, based on descriptor lists, respectively. A pipelining architecture also optimizes transfer of data between the buffers, the PCI bus, and the media access controller.

Brief Summary Text (17):
There is also a need for an arrangement enabling the use of a random access memory as a buffer in a network interface device, where potential synchronization problems between the clock domain of the host computer and the clock domain of the network are resolved to enable efficient control of the random access memory during the writing and reading of transmit or receive data.

Brief Summary Text (18):
There is also a need for an arrangement in a network interface device, where a synchronization circuit controls priority between writing and reading operations to and from the random access memory to enable efficient memory management for monitoring the status of stored frame data.

Brief Summary Text (19):
There is also a need for an arrangement in a network interface device, where a network interface architecture asynchronously monitors the status of data stored in a random access transmit buffer and a random access receive buffer to enable multiple memory controllers to store and read data into the random access memories using multiple clock domains.

Detailed Description Text (8):
The network interface device 10 also includes a buffer management unit 24 configured for managing DMA transfers via the DMA interface 16b. The buffer management unit 24 manages DMA transfers based on DMA descriptors in host memory that specify start address, length, etc. The buffer management unit 24 initiates a DMA read from system memory into the transmit buffer 18b by issuing an instruction to the DMA interface 16b, which translates the instructions into PCI bus cycles.

Hence, the buffer management unit 24 contains descriptor management for DMA transfers, as well as pointers associated with storing and reading data from the memory portion 18. Although the buffer management unit 24 and the memory management unit 22 are shown as discrete components, the two units may be integrated to form a memory management unit controlling all transfers of data to and from the memory unit 18.

Detailed Description Text (20):
The presence of two separate clock domains 56a and 56b in writing and reading to a random access memory 18 requires that the write controller and read controller devices be coordinated and synchronized to ensure that no contention issues arise due to the relative independence of the two clock domains 56a and 56b. The SRAM MMU 22 includes a synchronization circuit 60 that asynchronously monitors the status of the RX.sub.-- SRAM 18a and the TX.sub.-- SRAM 18b, enabling the memory management blocks 22a, 22b, 22c, and 22d to read and write to the memory 18 between the two clock domains 56a and 56b. Thus, problems that would ordinarily arise between the three clock domains (RMCLK, XMCLK, BCLK) in the individual memory management units 22a, 22b, 22c and 22d are avoided by use of the synchronization circuit 60 according to a prescribed arbitration logic.

Detailed Description Text (22):
FIG. 5A is a diagram illustrating multiple data frames (F1, F2, etc.) stored in the RX.sub.-- SRAM 18a. Assume that the RM.sub.-- MMU 22d is writing a sequence of data frames 64 (frame 1, frame 2, etc.) into RX.sub.-- SRAM 18a using a write pointer (WP), while the read controller 22c is reading out the data frames from the RX.sub.-- SRAM 18a to the BIU 16 using a read pointer (RP). If the read controller discards (e.g., flushes) a transmit data frame and desires to jump to the beginning of the next data frame, the synchronization circuit 60 must be able to track the start and beginning of each data frame to ensure that the read controller 22c properly locates the beginning of the next data frame.

Detailed Description Text (23):
FIG. 4 is a block diagram illustrating in detail the MMU 22. The synchronization circuit 60 includes asynchronous monitors 82a and 82b for asynchronously monitoring the amount of stored receive data and transmit data in the SRAMs 18a and 18b respectively, enabling the memory management units 22a, 22b, 22c, and 22d to track the number of stored data frames in their respective clock domains. The memory management unit tracks the number of stored data frames based on the difference between the write frame counter value and the read frame counter values. For example, the management block 22a includes a write counter 84a for incrementing a write pointer value (Tx Write Ptr.) in response writing the transmit data to the random access transmit buffer 18b. The management block 22b includes a read counter 84b for incrementing a read pointer value (Tx Read Ptr.) in response to reading the transmit data from the random access transmit buffer 18b. Similarly, the management block 22d includes a write counter 84d for incrementing a write pointer value (Rx Write Ptr.) in response to writing the receive data to the random access receive buffer 18a, and the management block 22c includes a read counter 84c for incrementing a read pointer value (Rx Read Ptr.) in response to reading the receive data from the random access receive buffer 18a. As described above, the XB.sub.-- MMU 22a and the RB.sub.-- MMU 22c operate in the PCI bus clock domain (BCLK) 56a, whereas the XM.sub.-- MMU 22b and the RM.sub.-- MMU 22d operate in the network clock domain 56b. More specifically, the XM.sub.-- MMU 22b operates in the network transmit clock domain (XMCLK), and the RM.sub.-- MMU operates in the network receive clock domain (RMCLK).

Detailed Description Text (25):
For example, assume a read frame counter pointer value (Rx Read Ptr.) and a write counter pointer value (Rx Write Ptr.) are stored in binary counters, where a write counter has a value (WR=100) and a read counter in the second independent clock domain transitions from (RD=011) to (RD=100). Since the clock domains 56a and 56b

operate independently of each other, a logic comparator performing a comparison between the write counter and read counter may erroneously conclude that the read and write counters have different values at a point in time where the read counter has a transitional value (e.g., 101, 111, or 000) as the read counter is being updated. Hence, the attempt to perform an asynchronous comparison between the binary read and write counters may cause an erroneous conclusion that the read and write pointers are not equal.

Detailed Description Text (28):
According to the disclosed embodiment, each management block 22a, 22b, 22c, 22d includes a counter 84 for the corresponding SRAM, where each counter 84 is configured for counting a pointer value by changing a single bit of a pointer value in response to a corresponding reading or writing operation.

Detailed Description Text (29):
As shown in FIG. 4, the write controller (RM.sub.-- MMU) 22d for the receive SRAM 18a is configured for writing a frame (e.g., the data frame 64) into the receive SRAM 18a according to a receive MAC clock (RMCLK) 74 synchronized relative to the network clock domain 56b. The write controller 22d, upon writing an entire data frame 64 into the receive SRAM 18a, outputs a write signal (RM.sub.-- ENF) to the synchronization circuit 60 indicating the completed writing of the data frame 64. Specifically, the write controller 22d writes the data frame 64 in FIG. 3 by receiving the frame data 68 from the MAC 20 according to the RMCLK 74. The write controller 22d then reserves a portion (location "X" in FIG. 5B) of the prescribed memory location 64 of the transmit SRAM 18a by first writing null data for the frame track 66 to hold header information. The actual frame data 68 is then written (location "Y"), followed by control information 70 (at location "Z"). Following the writing of the control information 70 at location "Z", the write controller 22d then returns to the frame track field 66 at location "X" and updates the frame track with the appropriate header information, including setting the end of frame address (ENF ADDR), updating the count (CNT) field indicating the number of DWORDS in the frame, the frame bit (FRM) indicating that the memory location 64 stores valid data, and the ENF bit which indicates the last double word of data. The write controller 22d concurrently supplies the frame track information (RM.sub.-- FRM.sub.-- TRK) to the synchronization circuit 60 for storage in a holding resistor 86a.

Detailed Description Text (30):
Hence, the write controller 22d outputs the write signal to the synchronization circuit 60 after updating the frame track field 66, at which point the SRAM 18a stores a valid data frame 64 at a prescribed location. As shown in FIG. 5A, successive writing of frames results in outputting a corresponding plurality of frame track values (RM.sub.-- FRM.sub.-- TRK) to the holding register 86a. Assuming, however, that the read controller 22c attempted to read the first frame F1 from the memory location 64 prior to completion by the write controller 22d, as shown in FIG. 5B, the read controller 22c would obtain invalid data, since the frame track field 66 would not yet be updated with valid data. In addition, the read controller 22c supplies the read frame track information from the accessed memory location (RB.sub.-- FRM.sub.-- TRK) to indicate the read status for the holding register 86a. Hence, invalid frame track data would be written to the holding register 86a if the read controller attempted a read operation before completion of the corresponding write operation.

Detailed Description Text (31):
According to the disclosed embodiment, the synchronization circuit 60 determines a presence of a complete stored data frame 64 in the random access memory 18a in response to read and write signals and independent of the bus clock 72 and the MAC clock (RMCLK) 74. Specifically, the synchronization circuit 60 includes a write receive frame counter (not shown) configured for changing a single bit of a write receive counter value in response to the write signal (RM.sub.-- ENF) output from

the write controller 22d. The synchronization circuit 60 also includes a read receive frame counter (not shown) configured for changing a single bit of a read receive frame counter value in response to the read signal (not shown) from the read controller 22c. As described above, the read controller 22c is configured for reading the frame 64 from the receive SRAM 18a according to a host bus clock (BCLK) 72, where the read controller 22c outputs a read signal to the synchronization circuit 60 in response to the reading of the frame.

Detailed Description Text (32):
According to the disclosed embodiment, the write receive frame counter and the read receive frame counter in the synchronization circuit 60 are implemented as gray code counters, such that only a single bit of a counter value is changed in response to assertion of the status signal from the corresponding memory controller 22. Use of the gray code counter ensures that any asynchronous comparison between the write receive frame counter and the read receive frame counter does not result in any erroneous values due to multiple bit transitions that may otherwise occur in counters using binary-format representations. The gray code counter values are compared asynchronously by the synchronization circuit, which outputs a one or more receive frame signal (RX.sub.-- FRM.sub.-- ONEM) indicating the presence of at least one stored receive frame in the receive SRAM 18a if the receive frame counter values are unequal.

Detailed Description Text (33):
The above-described arrangement is also used by the XB.sub.-- MMU 22a and the XM.sub.-- MMU 22b, where the XB.sub.-- MMU 22a outputs a signal (not shown) to the synchronization circuit 60 in response to writing a complete data frame into the TX.sub.-- SRAM 18b and the XM.sub.-- MMU 22b outputs a signal (not shown) to the synchronization circuit 60 in response to reading a complete data frame from the transmit SRAM 18b. The synchronization circuit includes a write transmit frame counter and a read transmit frame counter (not shown), implemented as gray code counters, to asynchronously determine the presence of at least one stored transmit frame in the transmit SRAM 18b. The synchronization circuit 60 outputs a one or more transmit frame signal (TX.sub.-- FRM.sub.-- ONEM) in response to asynchronously detecting at least one stored transmit frame in the transmit SRAM 18b.

Detailed Description Text (41):
As shown in FIG. 4, each management block 22.sub.i includes a memory counter for incrementing a memory pointer value in response to reading or writing the corresponding data from the corresponding SRAM. For example, the XB.sub.-- MMU 22a includes a transmit write counter 84a for incrementing a write pointer value in response to writing a double word (DWORD) into a memory location of the 32-bit SRAM 18b. According to the disclosed embodiment, each of the counters 84 are implemented as gray code counters, where the counter changes a single bit of the corresponding pointer value in response to the writing or reading of the corresponding data. Hence, the XB.sub.-- MMU 22a increments the write counter 84a by changing a single bit of the TX Write Ptr. value in response to writing a DWORD of transmit data into the TX.sub.-- SRAM 18b. Similarly, the XM.sub.-- MMU 22b includes a read counter 84b that increments the read pointer value (TX Read Ptr.) by changing a single bit of the corresponding read point value in response to reading a DWORD from the random access transmit buffer 18b. As shown in FIG. 4, the RX write counter 84d and the RX read counter 84c operate in the same manner, i.e., by changing a single bit of the corresponding pointer value in response to writing or reading a DWORD from the RX.sub.-- SRAM 18a.

Detailed Description Text (42):
As shown in FIG. 4, each of the management blocks 22a, 22b, 22c, and 22d output the corresponding pointer value to the synchronization circuit 60. Specifically, the RX Write Ptr. value and the RX Read Ptr. value are supplied to an asynchronous monitor 82a that asynchronously determines the amount of stored receive data (e.g., on a

double word basis) in the RX.sub.-- SRAM 18a. For example, the monitor 82a may include a comparator for performing a gray code comparison of the read and write pointer values. Hence, the asynchronous comparator 82a asynchronously monitors the amount of data stored in the random access receive buffer 18a.

Detailed Description Text (57):
As shown in FIG. 8A, the bus interface unit 16 includes a D flip flop 210a that latches the data (XB.sub.-- DATA) from the AD signal path of the PCI bus 12 in response to the host bus clock (BCLK). The D flip flop 210a outputs the 32-bit XB.sub.-- DATA signal to the holding register 212b, which selectively latches the XB.sub.-- DATA in response to an XB.sub.-- ADV signal received from the BIU 16. According to the disclosed embodiment, the multiplexer of holding register 212b is a multiple-selection input multiplexer, such that byte packing may be performed on the received XB.sub.-- DATA. Hence, the data output by the holding register 212b (XB.sub.-- SRAM.sub.-- DATA) is aligned in the TX.sub.-- SRAM to avoid the presence of invalid data within the TX.sub.-- SRAM due to invalid byte lanes (i.e., "holes"). The holding register 212a outputs the write address pointer (XB.sub.-- ADDR), as controlled by the incrementer 214a, controlled by the XB.sub.-- MMU 22a. As shown in FIG. 8A, the supply of data from the PCI bus 12 is performed in the host bus (BCLK) domain. Hence, the XB.sub.-- ADV signal from the BIU 16 enables the XB.sub.-- MMU to increment the write pointer in holding register 212a, and to supply data to the dual-port TX.sub.-- SRAM 18b.

Detailed Description Text (59):
As shown in FIG. 8B, the incrementer 214b, under the control of the XM.sub.-- MMU 22b, controls the read address pointer stored in holding register 212d. The MAC 20 outputs an advance signal (XM.sub.-- ADV), used to increment the read pointer in holding register 212d and the read signal in holding register 212e. According to the disclosed embodiment, the MAC 20 is able to assert the XM.sub.-- ADV advance signal until the TX.sub.-- SRAM is empty, based on an empty flag TX.sub.-- SRAM.sub.-- registers 212f and 212g are part of the MAC 20.

Detailed Description Text (61):
FIG. 8D is a diagram illustrating in detail the data path for the RB.sub.-- MMU 22c. According to the disclosed embodiment, the RB.sub.-- MMU 22c outputs the data onto the AD signal path of the PCI bus 12 with zero wait states in a DMA transfer, based on the holding registers 212l, 212m, 212n, and the multiplexer 216. As shown in FIG. 8D, the advance signal RB.sub.-- ADV output by the BIU 16 is used to drive the read pointer stored in holding register 212j, and the read pointer stored in holding register 212k. As described above, the read pointer value stored in holding register 212j is controlled by the incrementer 214c. Hence, the RX.sub.-- SRAM 18a outputs a double word of data from a prescribed location pointed to by the read pointer value (RB.sub.-- ADDR) as a 32-bit signal RB.sub.-- SRAM.sub.-- DATA. The holding register 212l outputs the signal RB.sub.-- DATA based on the RB.sub.-- ADV signal and the bus clock (BCLK) signal. Although not shown, the holding register 212l also performs byte alignment relative to the PCI bus 12 based on byte alignment (BE.sub.-- L) signals received by the multiplexer 220 from the descriptor management 24. The holding register 212l outputs the RB.sub.-- DATA signals to the holding register 212m and the multiplexer 216. The holding register 212m is configured to resolve any wait states or bus termination conditions (e.g., disconnect, retry) that may occur on the PCI bus 12. The multiplexer 216 and the multiplexer of the holding register 212n are controlled by a state machine (not shown) in the bus interface unit.

CLAIMS:

4. The method of claim 1, wherein the asynchronously monitoring step comprises:

incrementing a write pointer value in response to the storing step;

incrementing a read pointer value in response to the outputting step; and

asynchronously comparing the write pointer value and the read pointer value to determine an amount of stored data in the random access transmit buffer.

20. The method of claim 17, wherein the asynchronously monitoring step comprises:

incrementing a write pointer value in response to the storing step;

incrementing a read pointer value in response to the outputting step; and

asynchronously comparing the write pointer value and the read pointer value to determine the amount of data stored in the random access receive buffer.

24. The network interface device of claim 23, wherein:

the first management block includes a write counter for incrementing a write pointer value in response to writing the transmit data to the random access transmit buffer;

the second management block includes a read counter for incrementing a read pointer value in response to reading the transmit data from the random access transmit buffer; and

the synchronization circuit asynchronously determines the amount of data stored in the random access transmit buffer based on the write pointer value and the read pointer value.

25. The network interface device of claim 24, wherein the write counter and the read counter each increment the corresponding pointer value by changing a single bit of the corresponding pointer value.